

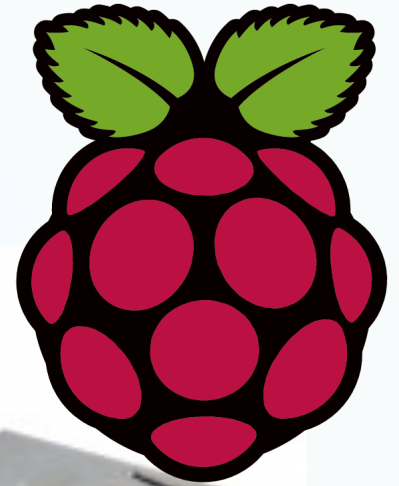
TRAC Program

Rachel Mroz

What have I been working on?

- Learning new ways to incorporate technology into my classroom
 - Investigating Raspberry Pi systems for my supervisor
 - Replace old high voltage controls with a new system and connections
 - Creating a program and GUI to test high voltage systems for an upcoming project

Raspberry Pi



Images from: <https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/>

GPIO Header

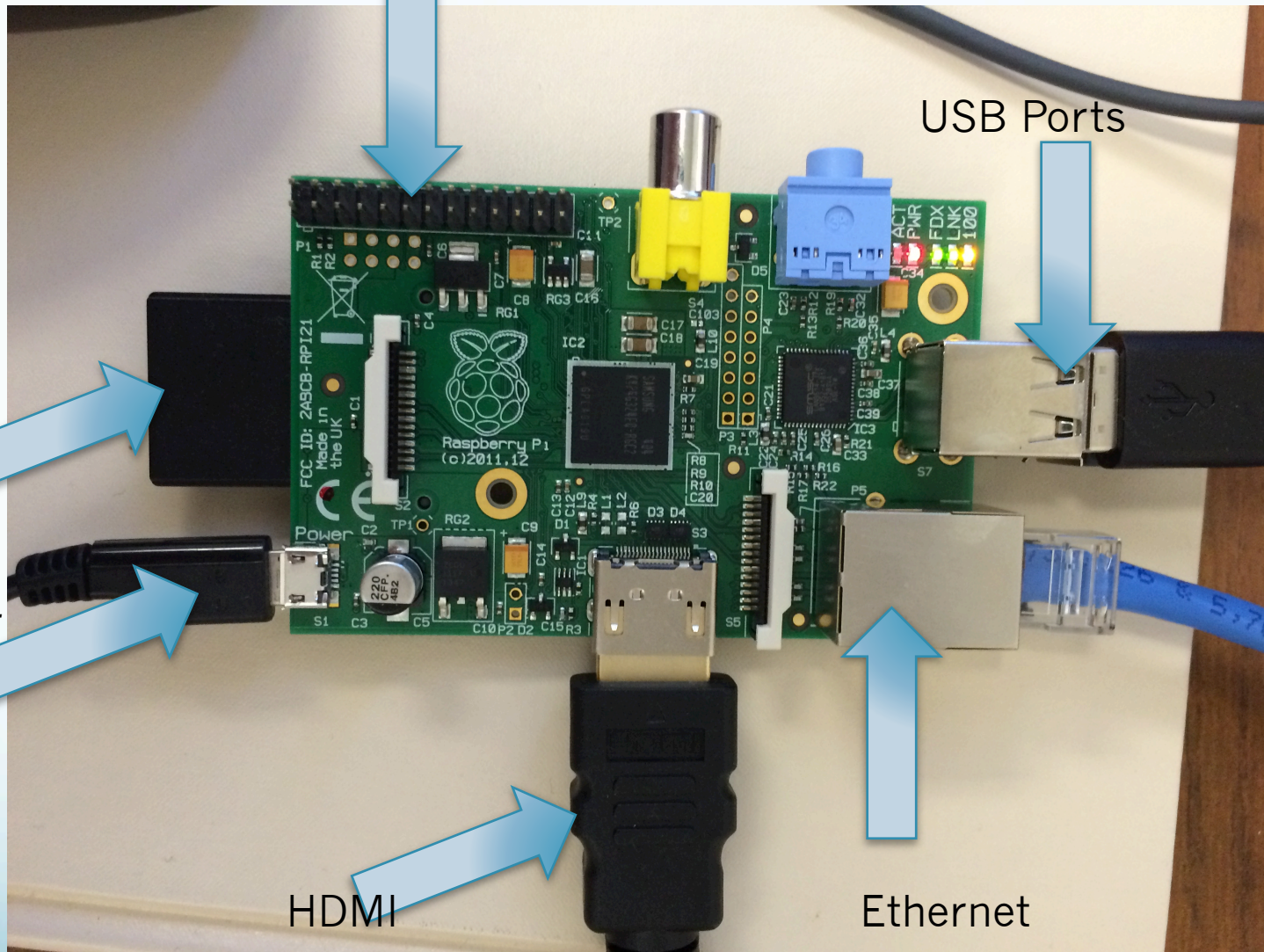
USB Ports

SD Card

Power

HDMI

Ethernet



Raspberry Pi

- Meant for students to learn about programming
- Uses Debian operating system
- Python is already installed

Raspberry Pi

- Small credit card sized computer intended to incorporate more computer science in the classroom
- All you need:
 - 5V charger
 - SD card
 - HDMI adapter
 - Monitor
 - Keyboard
 - Mouse

Applications

- Can be paired together with other Raspberry Pis
- Since it has a USB port can connect to many different devices
- Can also communicate with many different things because of the GPIO header
 - RS232 serial communications
 - LCD Screens
 - Touch screens

High Voltage Control System



Current High Voltage Control system

Really old Gateway Computer



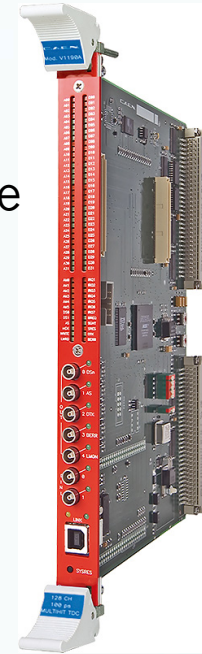
Gateway

High Voltage Power Supply System

Computer



Master Module



VME Module

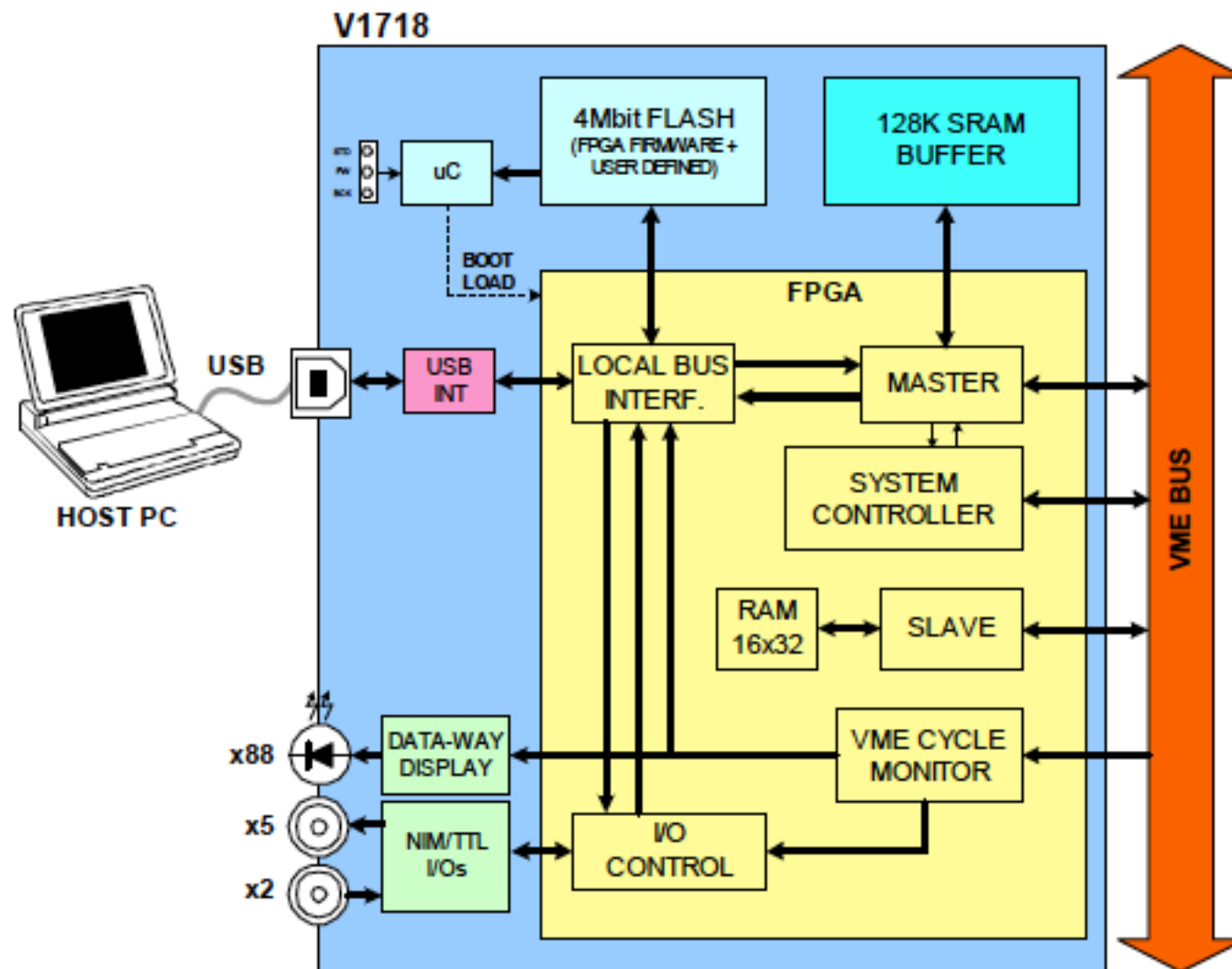


Detector



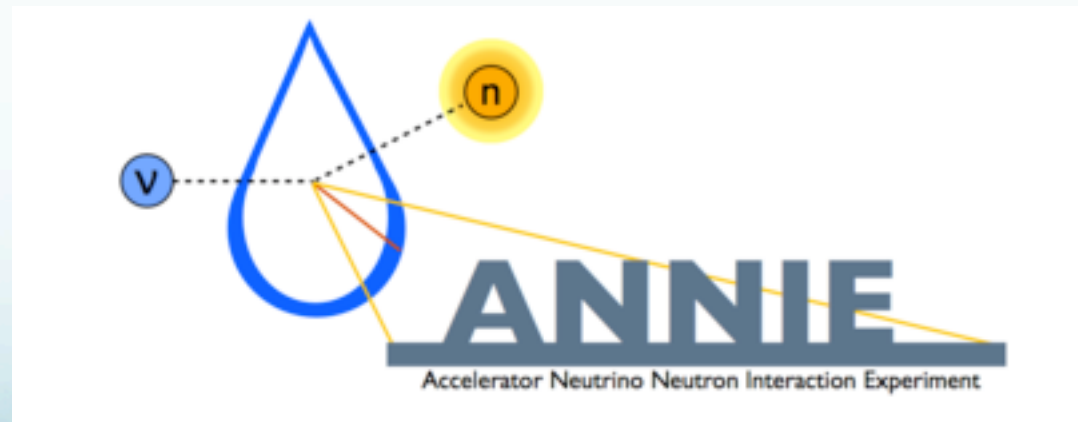
http://technofinancetrade.ru/tft/products/406/000/0000406/photo/_ .jpg

https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTzxfYGzeUe79IDT6Eo8mIMrA0rqt3TTvLIZuRu2k4Y-1_jcVII

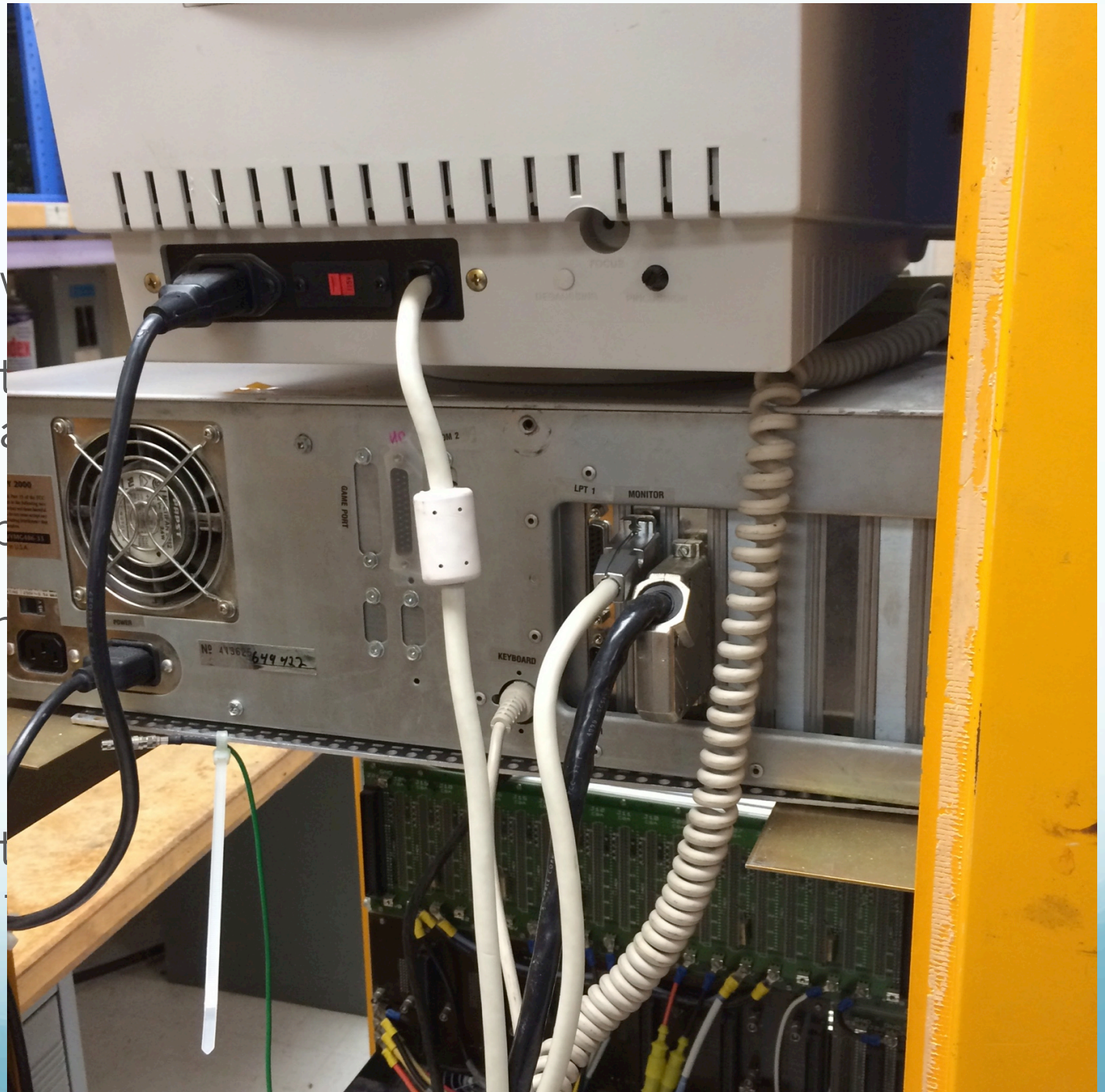


Why Bother?

- These supplies were used for the D-Zero experiment
- Why waste materials?
- An alternative to purchasing new materials for new experiments



- Provided v
- Figure out
Windows a
- Create a p
- Write pyth
 - Read
 - Writes
- Figure out
then put i



Providing a HV System

- Write a program that could read and write to various pods in a high voltage control system
- Challenges:
 - Needed to figure out the most efficient way possible
 - Never done this before
 - Understanding what the terms VME, reads, writes, bits, etc. mean ?!?!

Game Plan

- Understand what the demo program from CAEN did and what you needed to include
- Began writing the basic code in python with functions.
- Then put the functions into classes and developed a way to read and write and manipulate the code to get your data.
- It was a very step by step process and gradually became more and more difficult

rmroz@mnvdaqts-02:~/CAEN/sample

CAEN VME Manual Controller

R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS [EE000000]
2 - BASE ADDRESS [EE000000]
3 - DATA FORMAT [D16]
4 - ADDRESSING MODE [A32]
5 - BLOCK TRANSFER SIZE [256]
6 - AUTO INCREMENT ADDRESS [OFF]
7 - NUMBER OF CYCLES [1]
8 - VIEW BLT DATA

```
caen.py (~/.CAEN/sample/script) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Cut Copy Paste
caen.py x
#!/usr/bin/python
#
# Filename: caen.py
# Created : 06/23/2015
# Author  : Geoff Savage
#
# Wrap caen vem library in python.

from ctypes import *

# Set libFile to the correct directory and file name.
libFile = '/home/nfs/savage/CAEN/lib/x64/libCAENVME.so.2.41'

libc = cdll.LoadLibrary(libFile)
#print libc

swRel = create_string_buffer(20)
retVal = libc.CAENVME_SWRelease(byref(swRel))
print 'CAEN VME Library version =', swRel.value

cvSuccess = 0
cvV1718=0

bHandle = c_int()
bLink = c_int(0)
bDevice = c_int(0)
bBoard = c_int(cvV1718)

retVal = libc.CAENVME_Init(bBoard,bLink,bDevice,byref(bHandle))
if retVal:
    print 'Unable to contact VMEUSB board'

addr=c_int(0x8306)
val=c_int()
addrMod=c_int(0x20) # cvA16_S
dataWidth=c_int(0x02) # cvD16
retVal = libc.CAENVME_ReadCycle(bHandle,addr,byref(val),addrMod,dataWidth)
if retVal:
    print 'Unable to read address =', addr.value
else:
    print 'Value read from %s = %s' % (hex(addr.value), hex(val.value))

retVal = libc.CAENVME_End(bHandle)
if retVal:
    print 'Unable to close VMEUSB board'

Python Tab Width: 8 Ln 34, Col 12 INS
```

```
program1.py (~/.CAEN)
File Edit View Search To
Open Save
program1.py x
#!/usr/bin/python
#
#Filename: program1
#Created 06/24/2015
#Author: Rachel Mroz
#
#Read and Write to boards

from ctypes import *

# Set libFile to the correct path
libFile = '/home/nfs/rmroz/

libc = cdll.LoadLibrary(libFile)
#print libc

swRel = create_string_buffer(1024)
retVal = libc.CAENVME_SWRel(swRel, 1)
print 'CAEN VME Library version: %s' % swRel

cvSuccess = 0
cvV1718=0

bHandle = c_int()
bLink = c_int(0)
bDevice = c_int(0)
bBoard = c_int(cvV1718)

retVal = libc.CAENVME_Init(&bHandle, &bLink, &bDevice, &bBoard)
if retVal:
    print 'Unable to contact board'

def Motherboard(Address, function):
    if function== 1:
        g = 0xF000
    elif function== 2:
        g = 0x0000
    h = Address + str(4)
    j= '0x%s' % (h)
    Addr= int(j, 16)
    PowerCycle(Addr, g)

def PowerCycle(addr, write):
```



```
programClass.py (~ /CAEN/sample/script) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Cut Copy Paste Find
programClass.py x

class Crate:
    #determine number on crate
    def CrateAddress(self):
        b= input('Crate Number: ')
        Crate.Address = '0x%s000' % (b)
        return Crate.address

class Motherboard(Crate):
    def address(self):
        z = input('Module:')
        a = '0x%s00' % (z)
        Motherboard.address = Crate.address + int(a, 16)
        print Motherboard.address

class Module(Motherboard):
    def module(self):
        z = input('Pod: ')
        a = int('0x%s00' % (z) , 16)
        Module.address = Motherboard.address + a
        return Module.address

    def turnon(self):
        addr = c_int(Module.address + 0x4)
        data=c_int(0xF000)
        addrMod=c_int(0x2d)
        dataWidth=c_int(0x02)
        retVal = libc.CAENVME_WriteCycle(bHandle, addr, byref(data), addrMod, dataWidth)
        if retVal:
            print 'Cycle not completed'
        else:
            print 'Cycle completed'

    for a in range(00, 120, +0x10):
        Module.address= + a
        #Module.addr = hex(address)---- redundant???
        #Module.addr=int(Module.addr, 16)
        return Module.address

    def initialize(self):
        addr = c_int(Module.address + 0x4)
        data=c_int(0xF000)
        addrMod=c_int(0x2d)
        dataWidth=c_int(0x02)
```

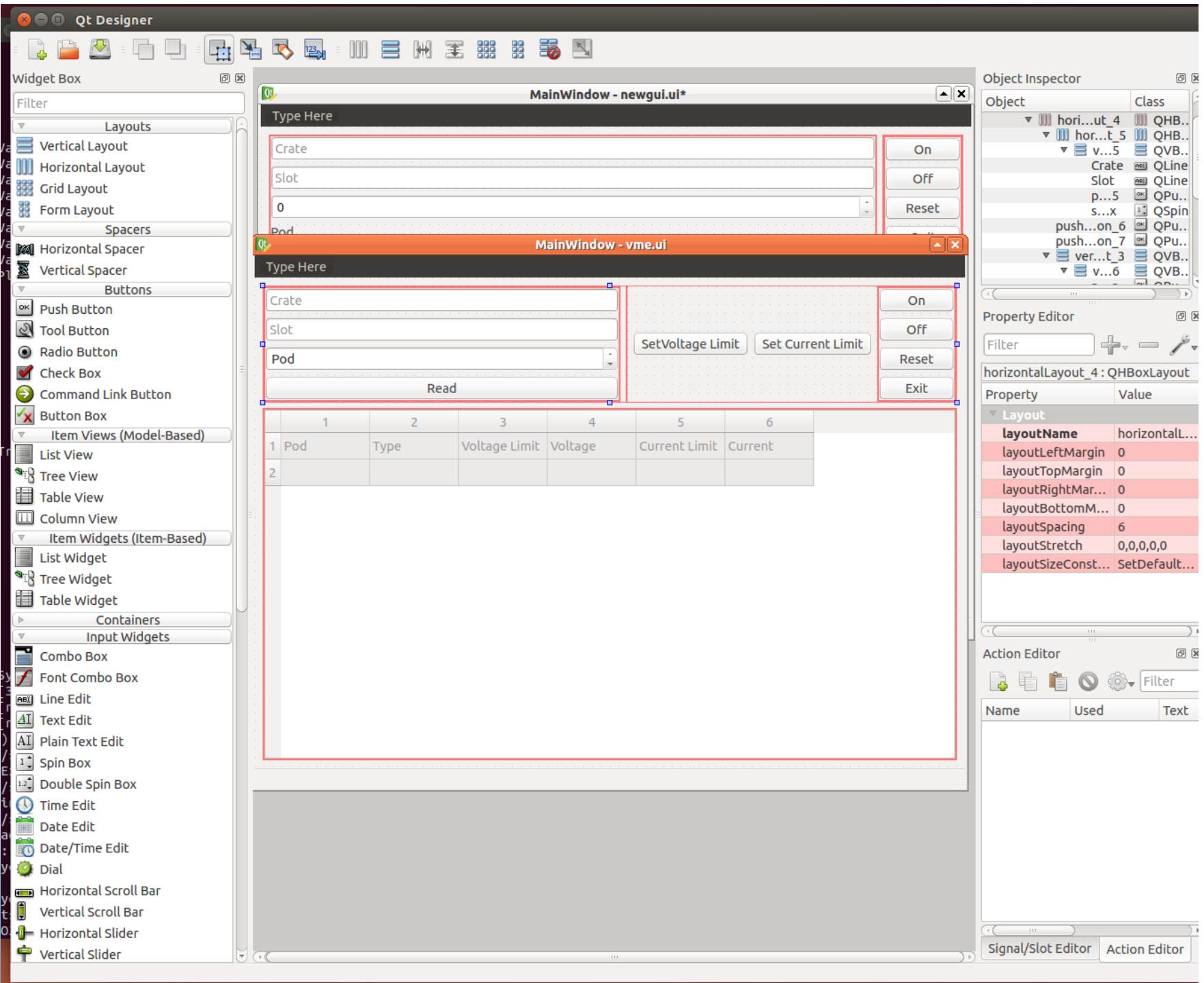
Python Tab Width: 8 Ln 96, Col 21 INS

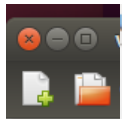
GUI

- After the script was written, I tested it for bugs and errors
- Began to make a Graphical User Interface (GUI) using PYQT.
- Again the same challenges with writing the python script

Crate	+5 Digital	+12 Analog	-12 Analog	+12 Bulk	-12 Bulk	Temp DegC
Pixel						
M217C	4.98	12.29	-12.24	12.18	-12.27	20.26
M217D	5.16	12.15	-12.42	12.48	-12.36	21.79
M217E	5.09			12.36	-12.36	20.52
M218C	5.05			12.24	-12.29	22.84
M218D	5.16			12.43	-12.30	24.12
M218E	5.11	12.32	-12.30	12.42	-12.27	22.81
Central						
M215B	5.04	12.68	-12.53	12.40	-12.44	24.28
M215C	5.10	12.35	-12.41	12.50	-12.47	22.86
M215D	5.05	12.31	-12.24	12.41	-12.52	24.41
M215E	5.17	12.21	-12.21	12.23	-12.25	26.53
M217B	5.04	12.26	-12.14	12.21	-12.02	26.81
M218B	4.95	12.35	-12.34	12.33	-12.38	22.54

**Bulk supply voltages
for current pump**





vmegui

```
#!/usr/
```

```
#
```

```
#Filename
```

```
#Created
```

```
#Author:
```

```
#make a
```

```
import s
```

```
from PyQ
```

```
class Ex
```

```
def
```

```
def
```

```
)
```

```
)
```

```
rat
```

```
NAL
```

```
ot', self)
```

```
AL('valueChanged(int)'), self.Motherboard)
```

VME Status

Crate

0

Set

On

Slot

0

Off

Pod

0

Reset

Read Type

Set Voltage

Set Current Limit

Quit

	Pod	Type	Volt limit	Voltage	Current Limit	Current
1						
2						
3						
4						
5						
6						
7						
8						
9						

Next Steps

- Finish connecting GUI to python script
- Run it with fake data and information to see errors



What I have learned

Applications

- There are many different things I have learned this summer
 - About how I learn and what I need to succeed
 - Trial and error is really important!
 - Not being afraid to ask questions

How to use in my classroom

- Created a raspberry pi and python resource sheet to be implemented into our computer science course

Raspberry Pi Tech Specs

Intro to Raspberry Pi - <https://www.raspberrypi.org/help/faqs/>

Raspberry Pi-- B and B+

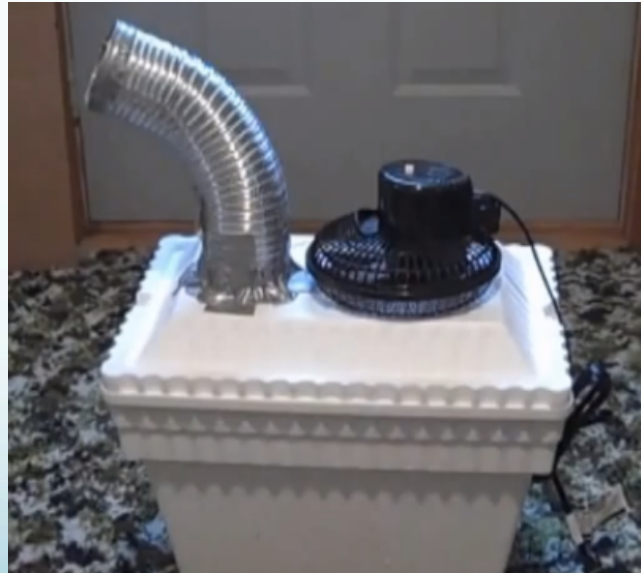
(Raspberry Pi- B- This one has the two usb ports, that is the easiest way to tell the difference)

Tech Specifications

	B	B+
SoC	Broadcom BCM2835	Broadcom BCM2835
CPU	700 MHz single-core ARM1176JZF-S	700 MHz single-core ARM1176JZF-S
GPU	Broadcom VideoCore IV @ 250 MHz, OpenGL ES 2.0 (24 GFLOPS) MPEG-2 and VC-1, <u>1080p 30</u> H.264/MPEG-4 AVC high	Broadcom VideoCore IV @ 250 MHz, OpenGL ES 2.0 (24 GFLOPS) MPEG-2 and VC-1, <u>1080p 30</u> H.264/MPEG-4 AVC high profile decoder and encoder

Create an Air Conditioner

- Create a small air conditioner that can be powered using simple supplies
- Create a python program that can control the temperature and tell the fan to turn on



Acknowledgments

- Thank you to
 - Geoff Savage, mentor
 - Harry Cheung, TRAC program head
 - Pratima Jindal, TRAC program
 - The other TRAC teachers

References

- http://cdorg.fnal.gov/ese/prep/catalog/hardware_info/bira/vme4877.html
- http://www-d0.fnal.gov/runcoor/DAQ/Tutorials/2004/2004-11-09_Bartlett_HV.pdf